1. ADO.NET + Windows Forms
2. EntityFramework + ModelFirst + WPF + MVVM

MVVM => Model – ViewModel-View

Классическая разделение проекта на 3 слоя

1. Слой 1 Model – напрямую отражает БД
2. Слой 2 VM – Бизнес Логика + отражение 1 слоя + взаимодействие с 3 слоем
3. Слой 3 – View – интерфейс юзера (кнопочки, менюшки и тд)

Благодаря слоям проще соблюдать принципы SOLID

Внутри MVVM важную роль играет паттерн “Команда” Command

Относится к группе “Поведенческих”

Цель: отделить “Заказчика Действия” от “исполнителя действия” за счет посредника – “Команда”

1)Наличие посредника делает связь между бизнес-логикой и интерфейсом максимально слабой.

2) Использование паттерна КОМАНДА позволяет красиво и удобно добавить функционал “история действии” и “отмена действии”

3) Если есть команда – можно легко добавить одинаковые действия в БЛ на разные элементы UI.

Важнейшая проблема разработки приложений – СИЛЬНАЯ связь между бизнес логикой и интерфейсом пользователей

В непрерывной разработке чаще всего приходится менять БЛ или UI

Для WPF этот паттерн писать не нужно поскольку необходимые интерфейсы и классы уже присутствуют

Для Windows forms нет нужно писать

UML – Unified Model Language